Properties of Regular Languages and FA: Complements and Intersections on Regular Languages and Finite Automata

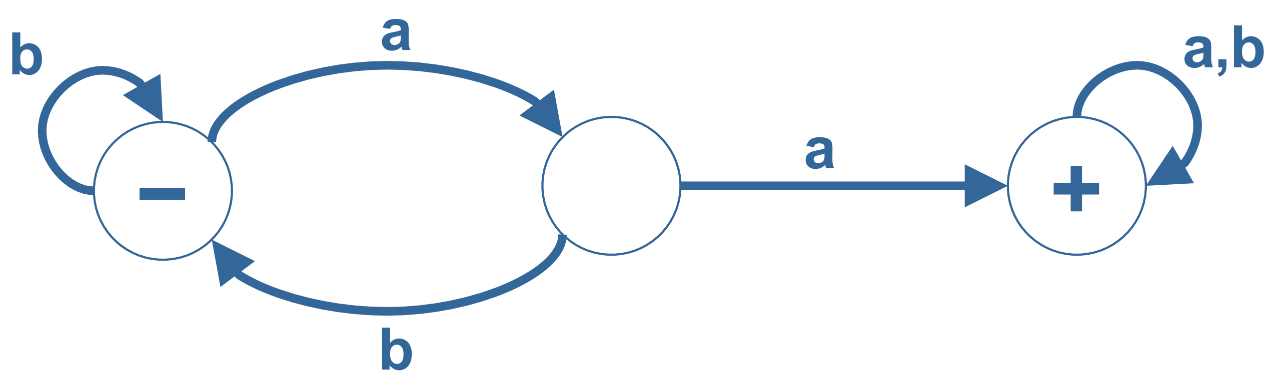
Oneil B. Victoriano

CS 3143 Automata Theory and Formal Languages

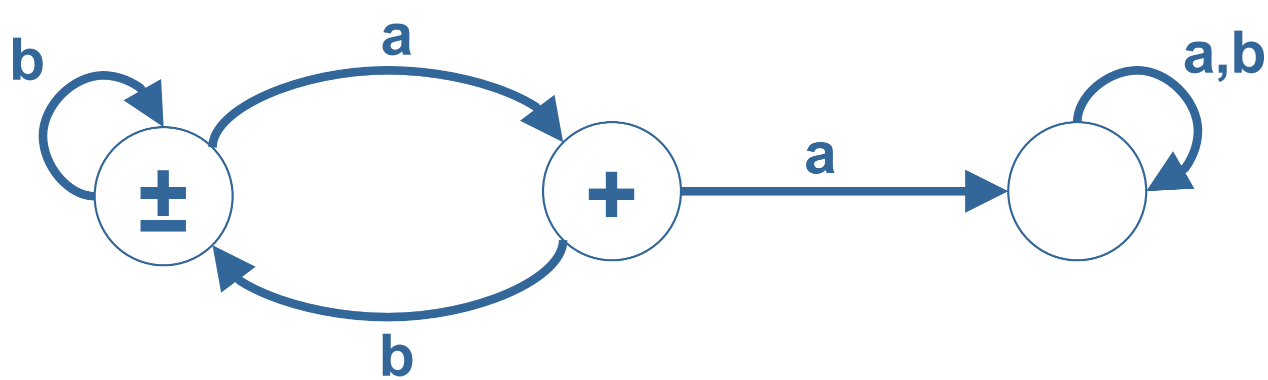
Complement

- If L is regular set over Σ, then Σ* – L is also regular over Σ.
- can be obtained by swapping its end states with its non-end states, vice-versa.
- leave the start state as is

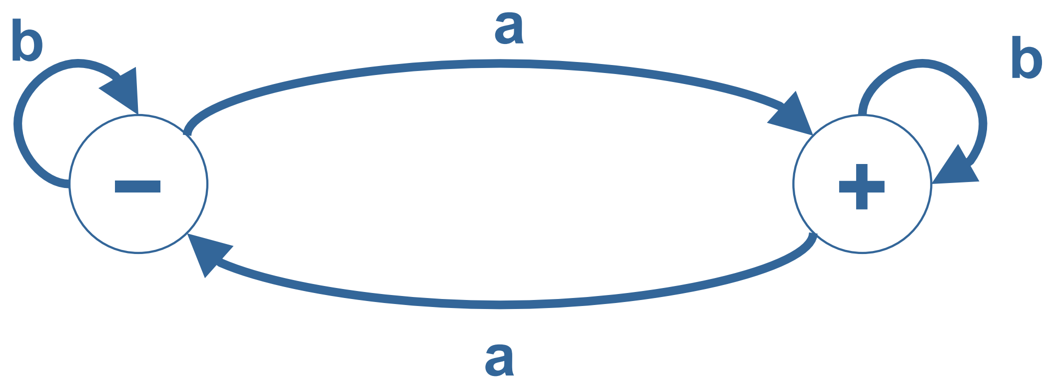
FA1



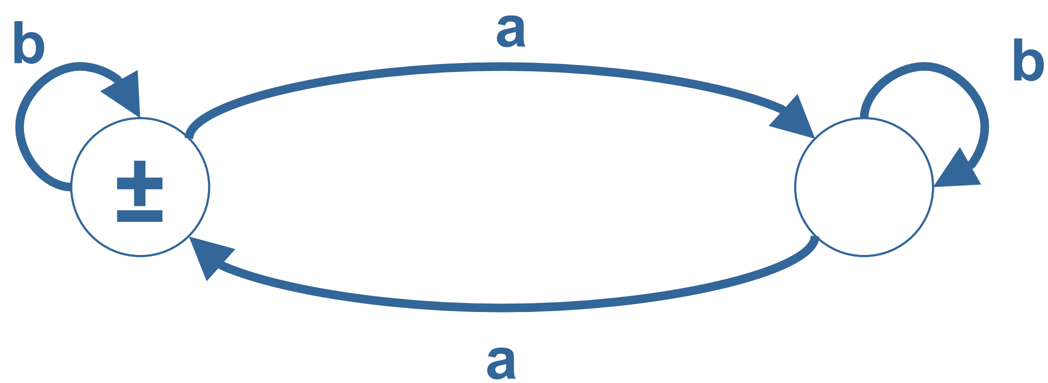
FA1’



FA2



FA2’



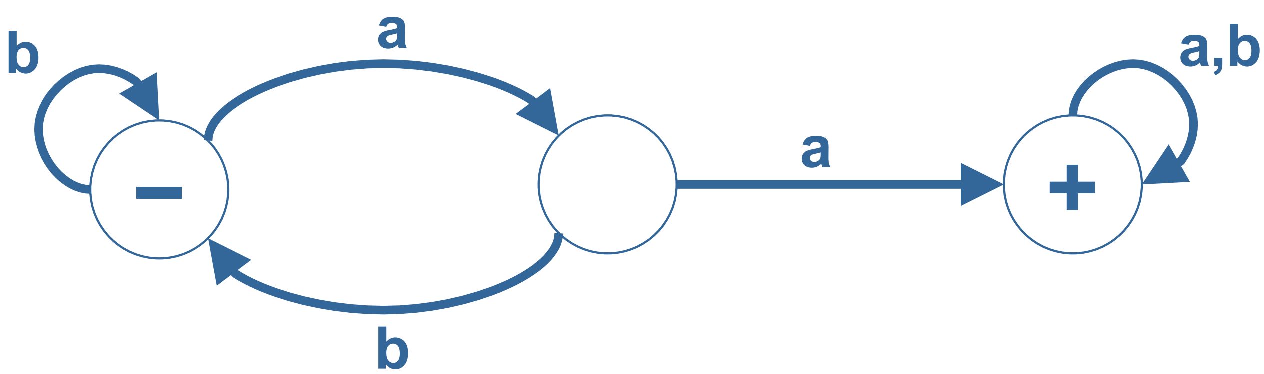
Intersection

- If L1 and L2 are regular languages, then $L1 \cap L2$ is also a regular language. In other words, the set of regular languages is closed under intersection.
- $L1 \cap L2 = (L1' + L2')'$
- Proof: by the DeMorgans' law for sets

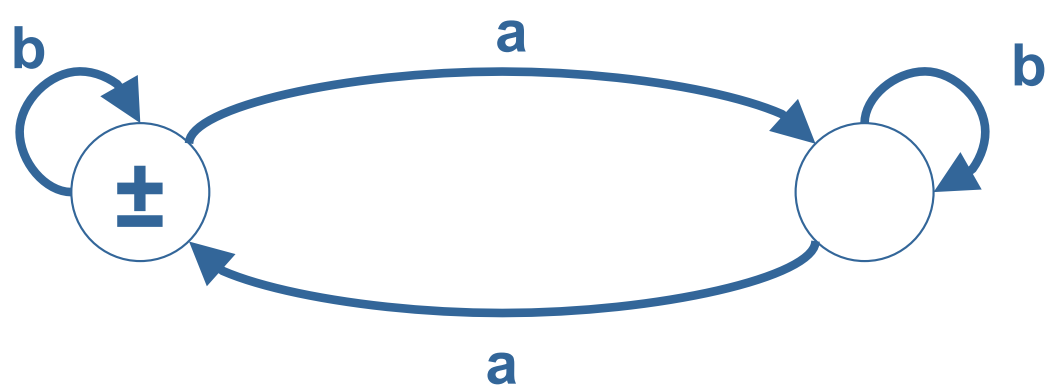
Example

- Two languages over Σ= {a, b}
- L1 = all strings with a double a
- L2 = all strings with an even number of a's
- Both are regular languages, bcoz they can be defined by the following regular expression
  - r1 = (a+b)*aa(a+b)*
  - r2 = b*(ab*ab*)*

FA1

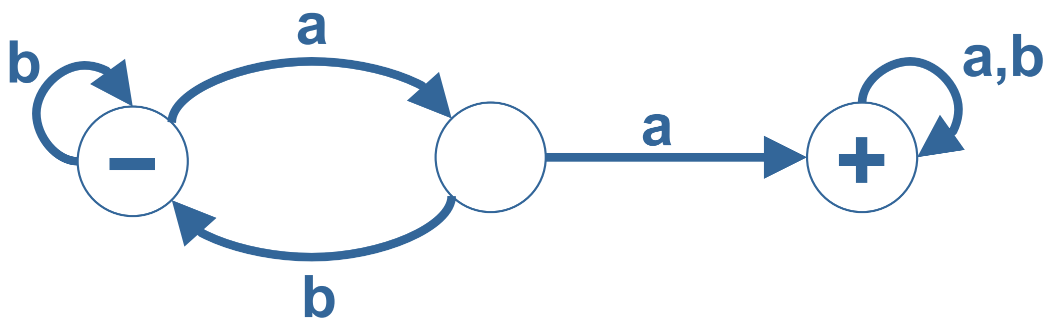


FA2

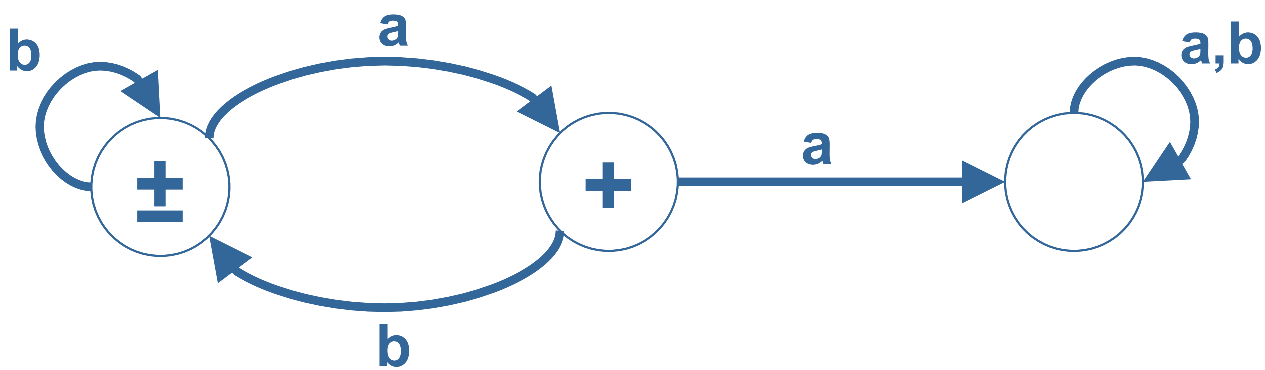


- The first step in building the machine (and regular expression) form L1 ∩ L2 is to find the machine that accept the complementary languages L1’ and L2’
- L1’ = all strings that do not contain the substring aa
- L2’ = all strings having an odd number of a’s

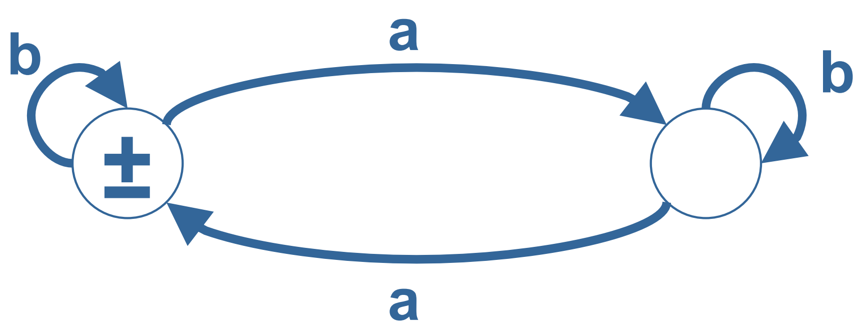
FA1



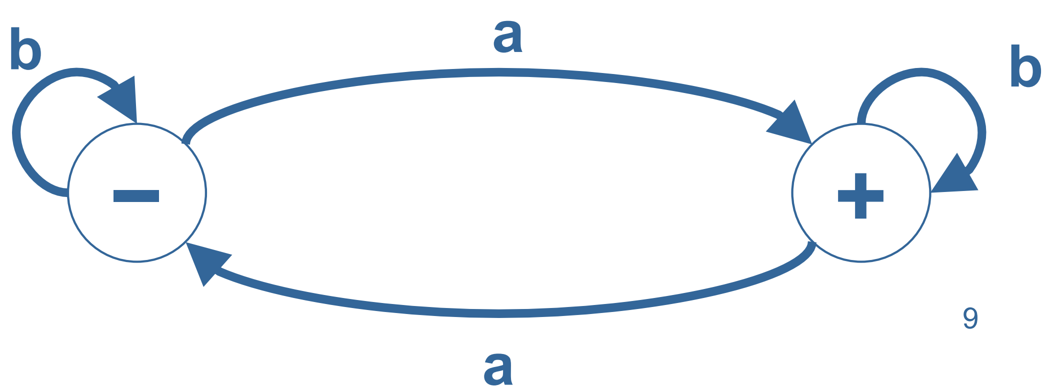
FA1’



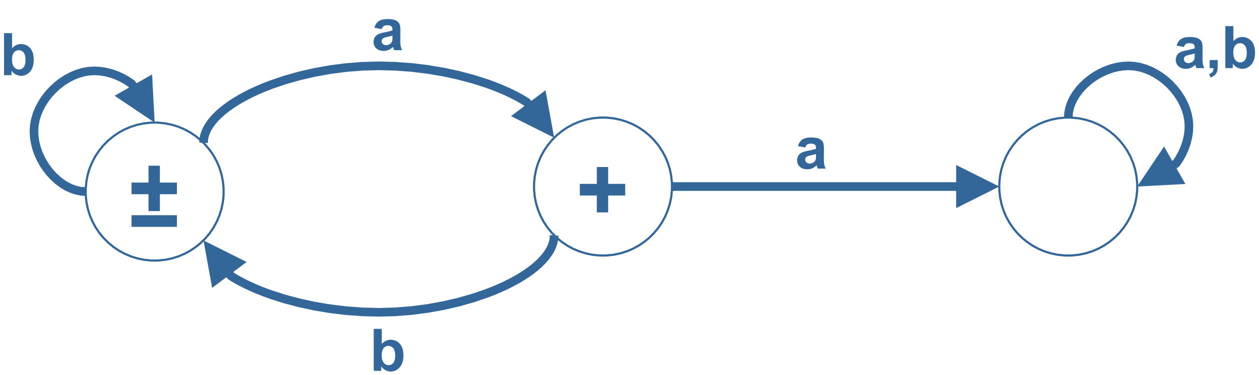
FA2

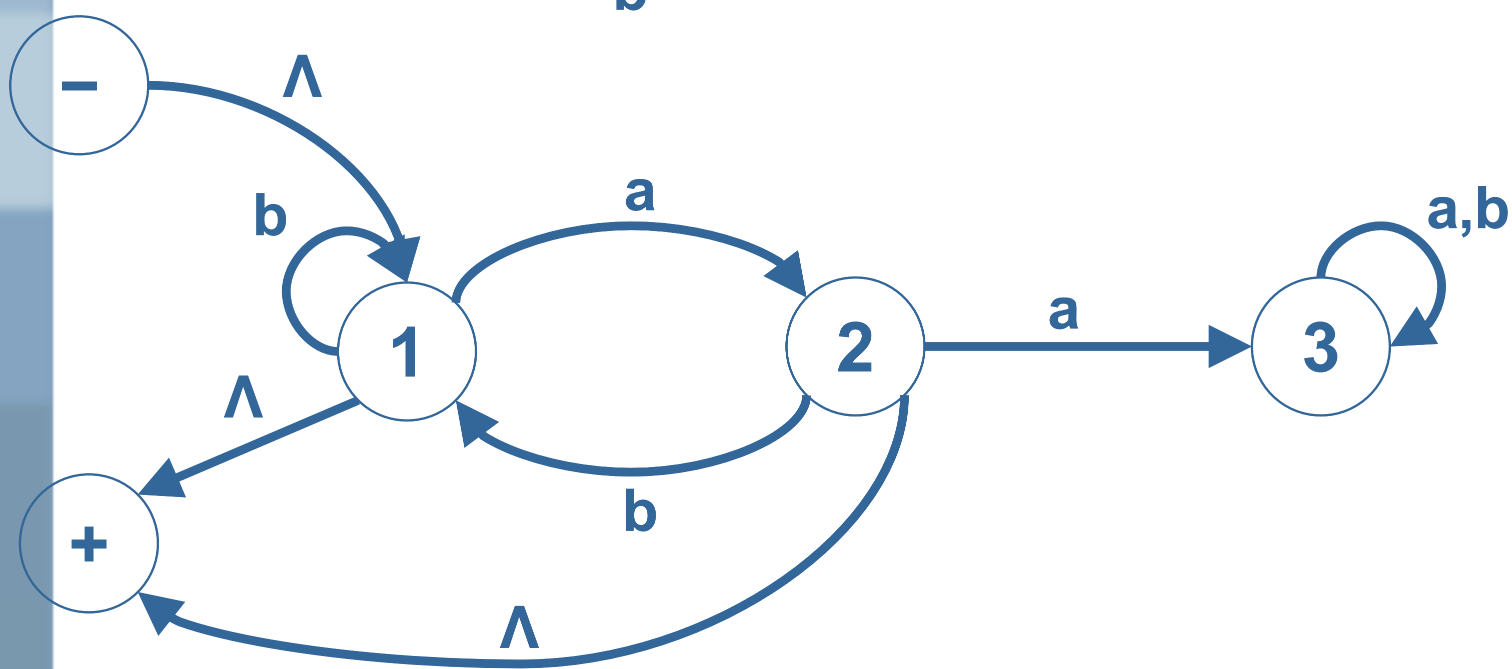


FA2’

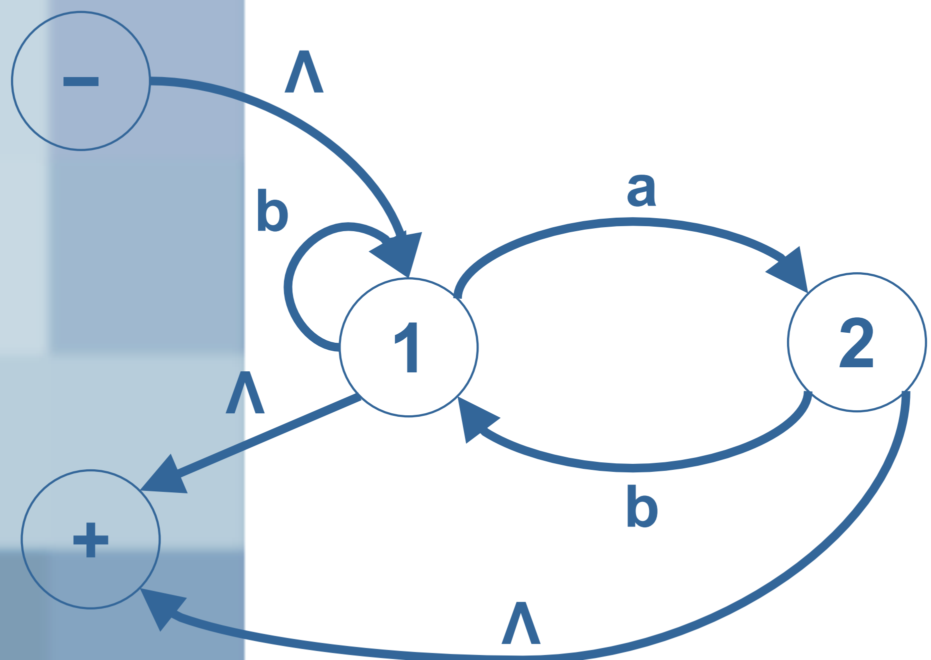


FA1’

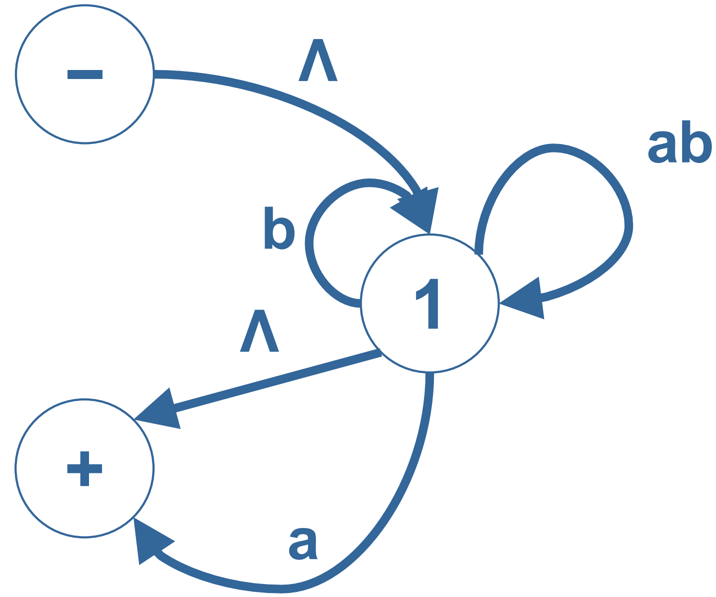




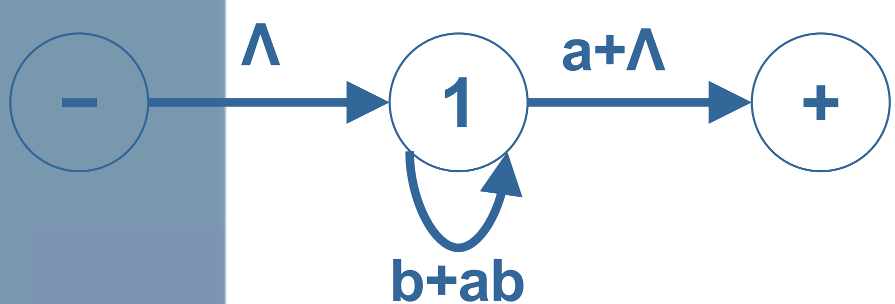
1. Dropped state 3



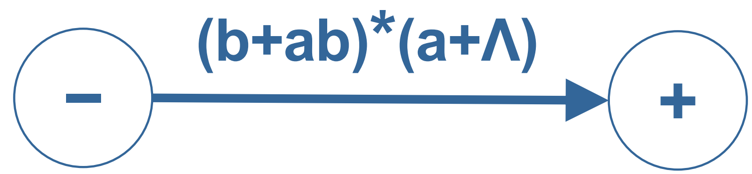
2. Bypass state 2



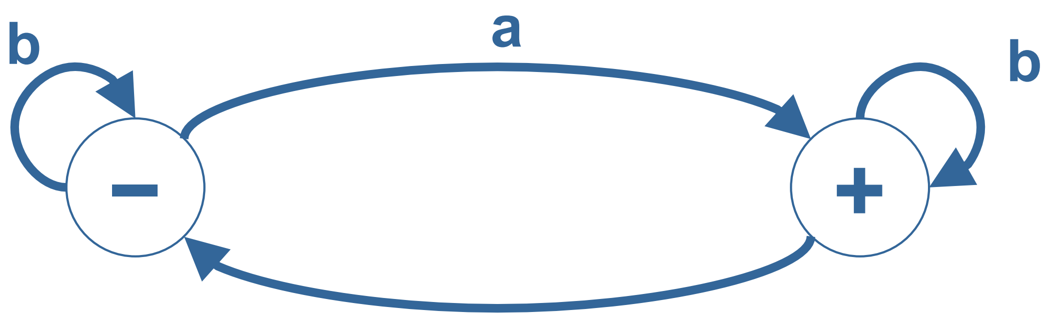
2. Bypass state 2

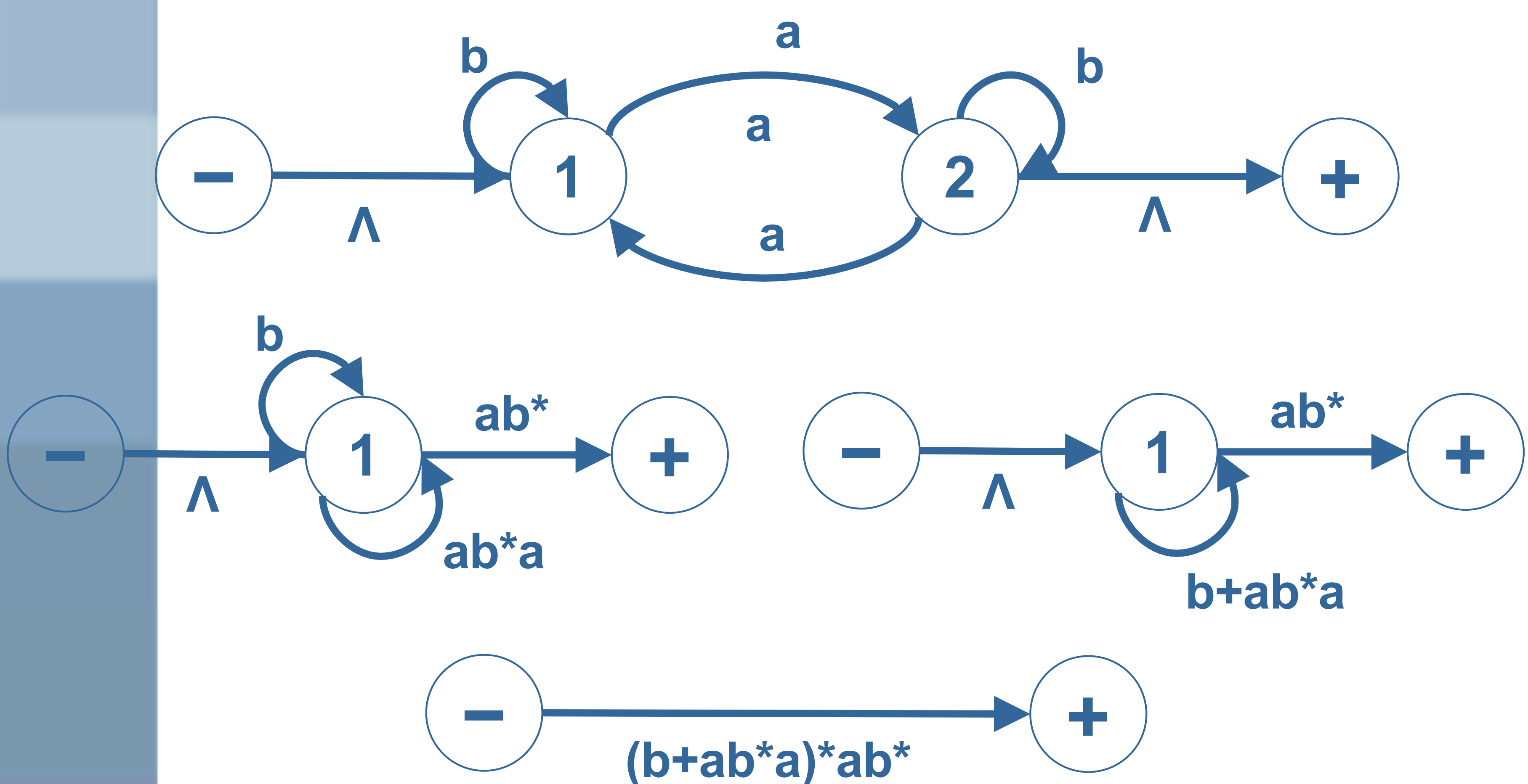


3. Bypass state 1



FA2’





- regular expression for L1’ + L2’

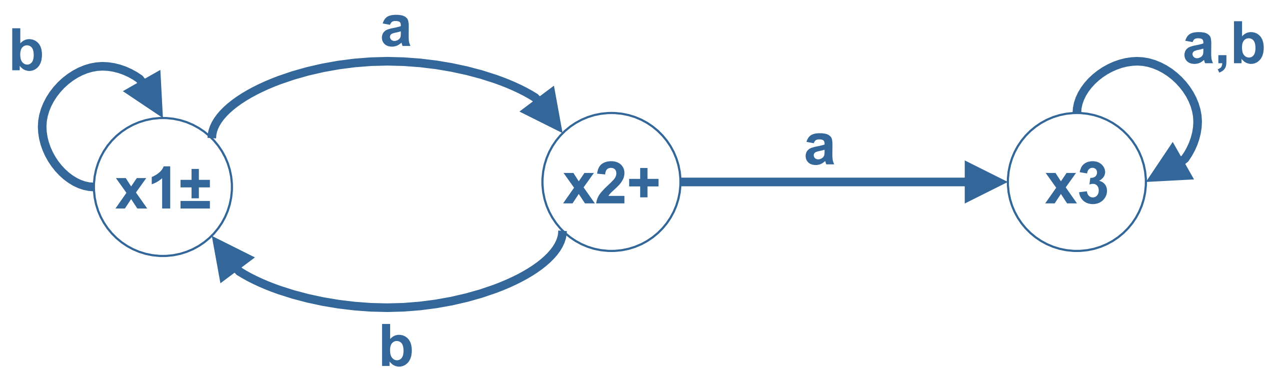
$$r1' + r2' = (b+ab)^*(\Lambda+a) + (b+ab^*a)^*ab^*$$

- Make this regular expression into an FA so that we can take its compliment to get the FA that defines intersection of two FAs.

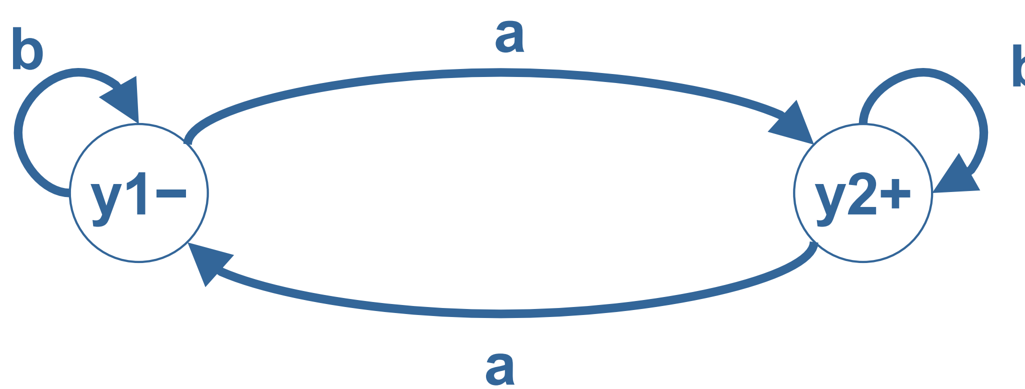
$$L1 \cap L2 = (r1' + r2')'$$

- Alternative approach is to make the machine L1’ + L2’ directly from the machines for L1’ and L2’ without resorting to regular expressions.

FA1’



FA2’



Where the start states are x1 and y1, and the final states are x1, x2, and y2.

- The six possible combinations states are

z1= x1 or y1 start, final (words ending here are accepted on FA1')

z2= x1 or y2 final (words ending here are accepted on FA1' & FA2')

z3= x2 or y1 final (words ending here are accepted on FA1')

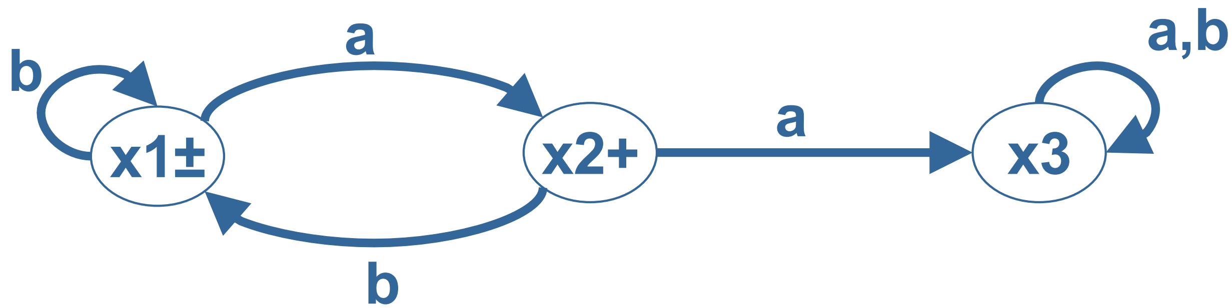
z4= x2 or y2 final (words ending here are accepted on FA1' & FA2')

z5= x3 or y1 not final or either machine

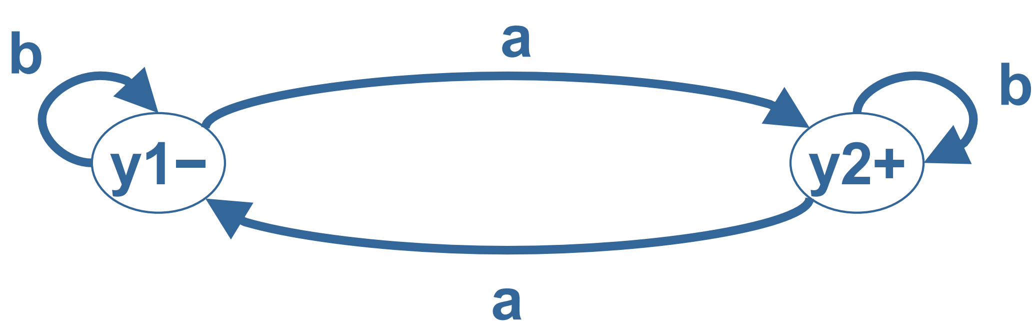
z6= x3 or y2 final (words ending here are accepted on FA2')

Z	x	y	FA1’	FA2’	FA1’+F2’	(FA1’+F2’)’	x	y	xy	x	y	xy
z1	x1	y1	- +	-	- +	-	x2	y2	z4	x1	y1	z1
z2	x1	y2	- +	+	+		x2	y1	z3	x1	y2	z2
z3	x2	y1	+	-	+		x3	y2	z6	x1	y1	z1
z4	x2	y2	+	+	+		x3	y1	z5	x1	y2	z2
z5	x3	y1		-		+	x3	y2	z6	x3	y1	z5
z6	x3	y2		+	+		x3	y1	z5	x3	y2	z6

FA1’



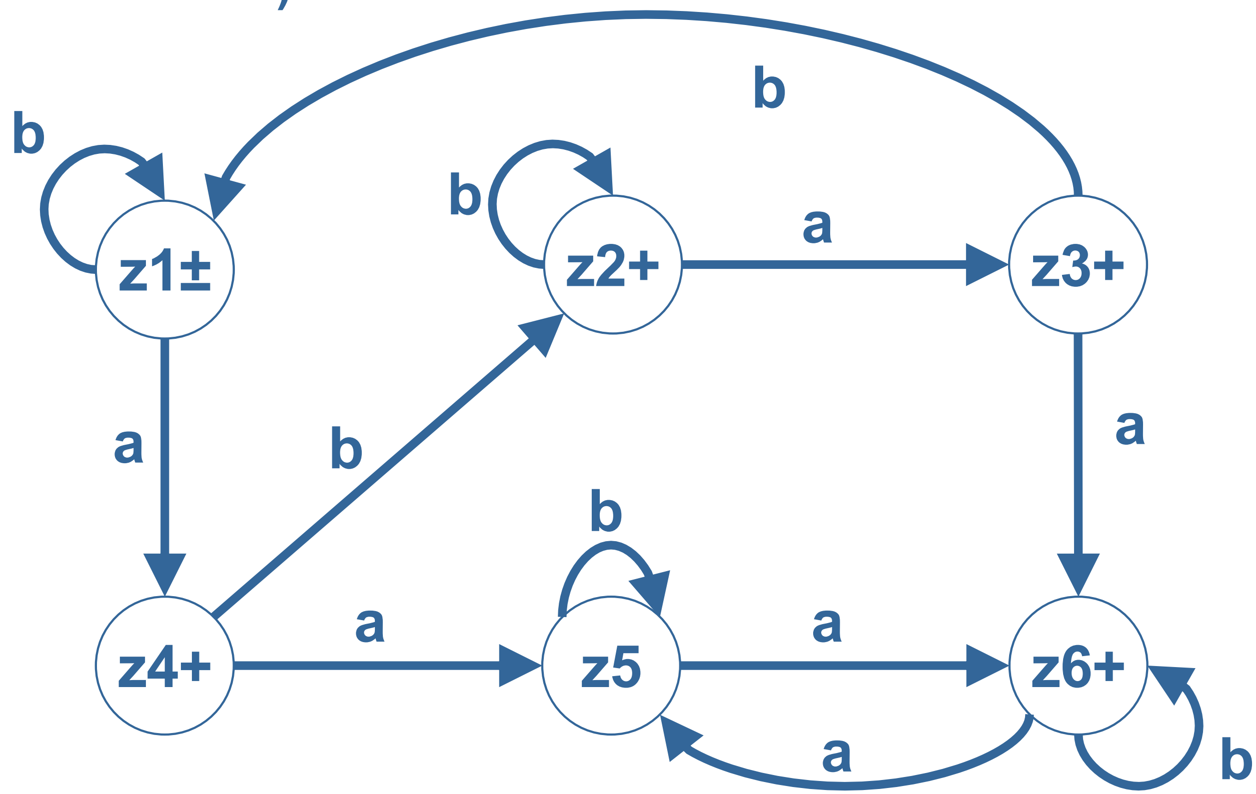
FA2’



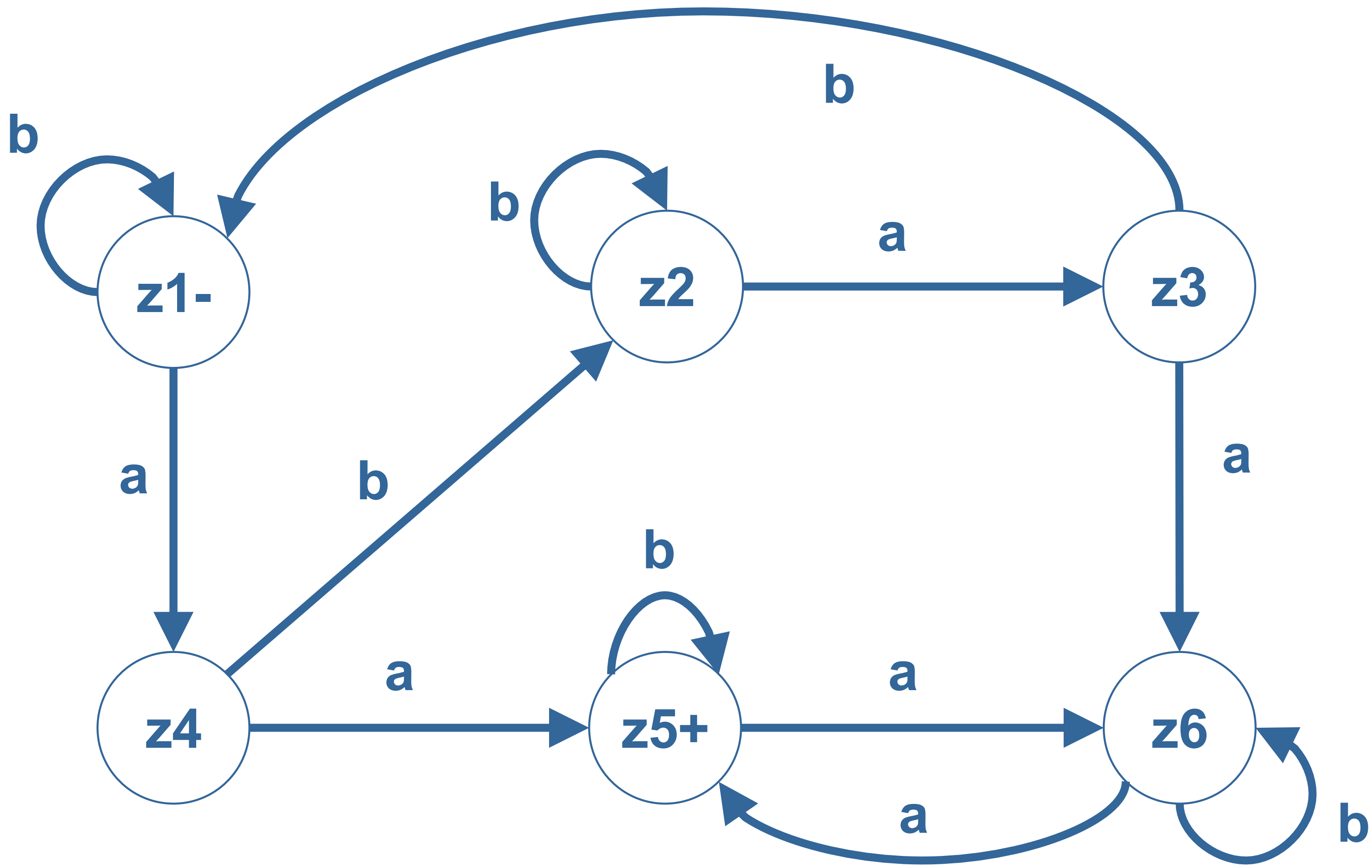
Transition table & graph

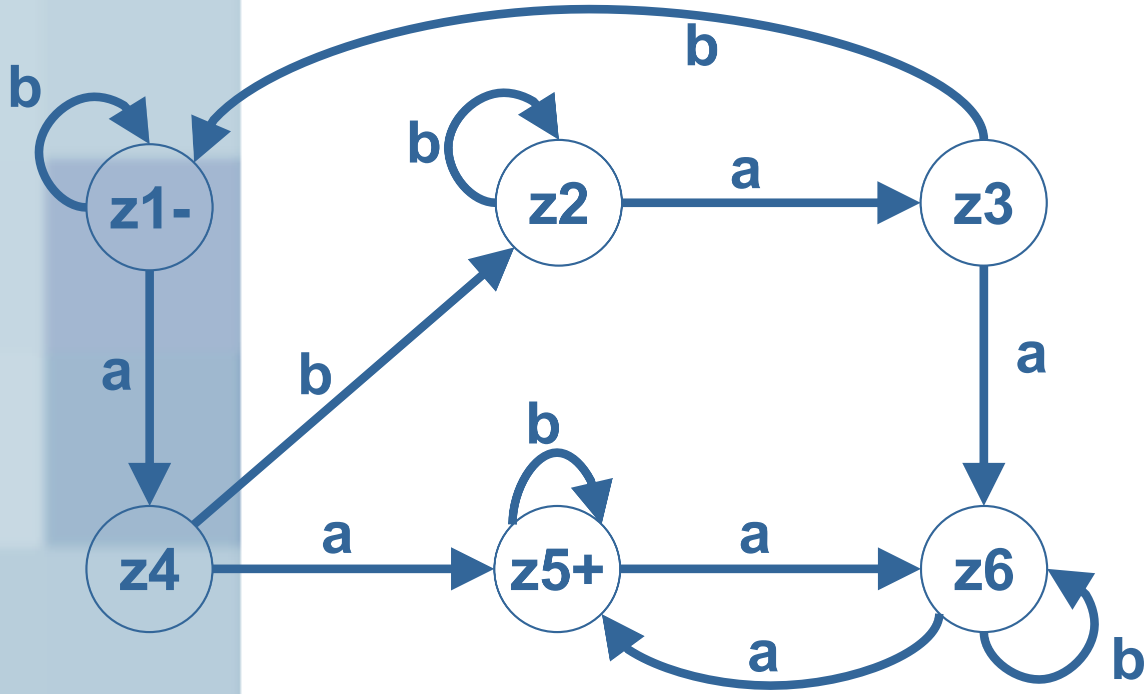
(FA1' + FA2')

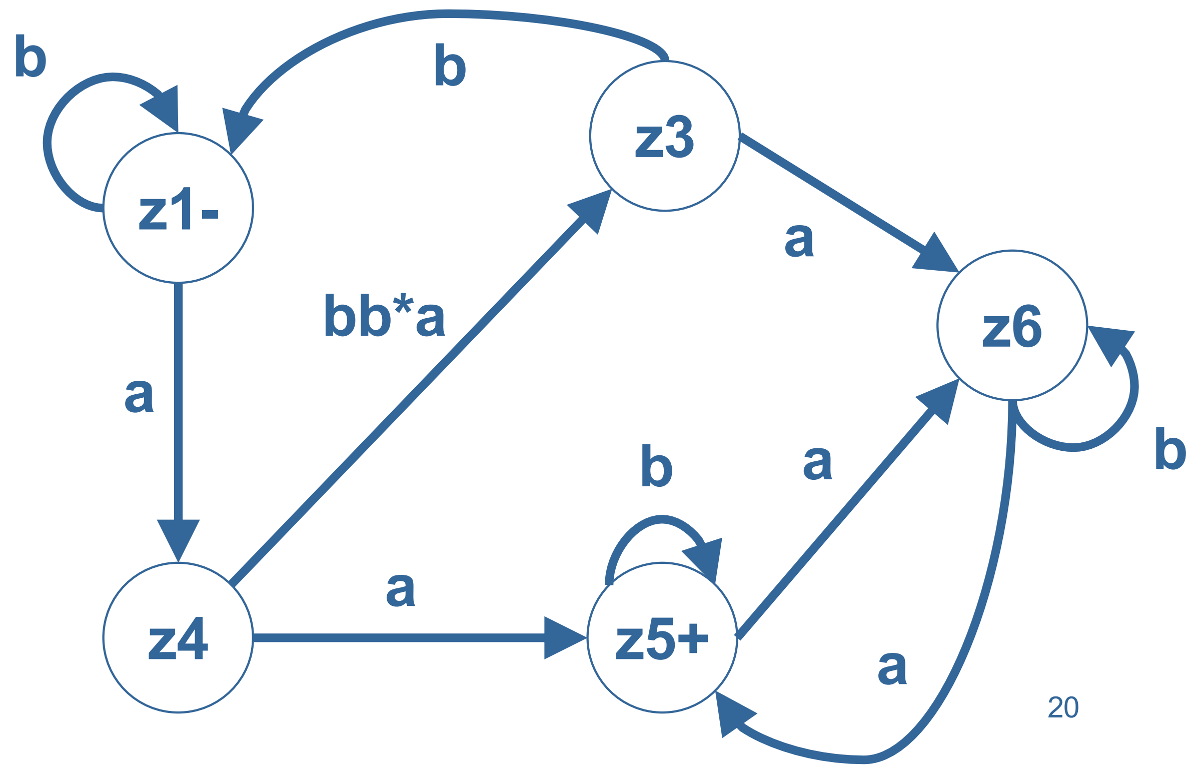
	a	*b*
± z1	z4	z1
+ z2	z3	z2
+ z3	z6	z1
+ z4	z5	z2
z5	z6	z5
+ z6	z5	z6

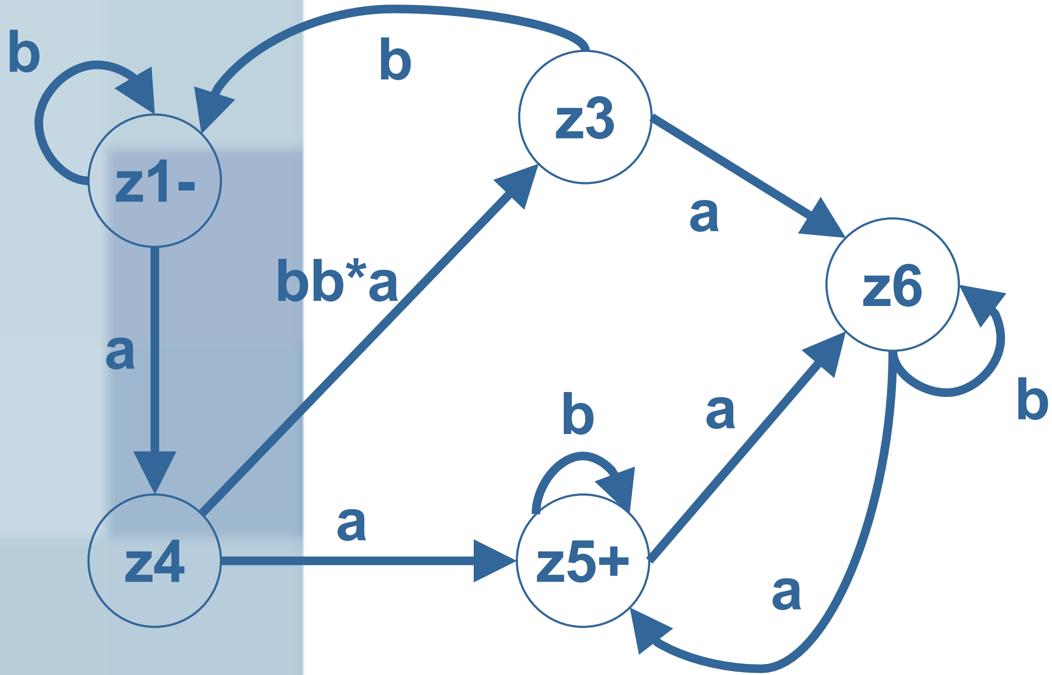


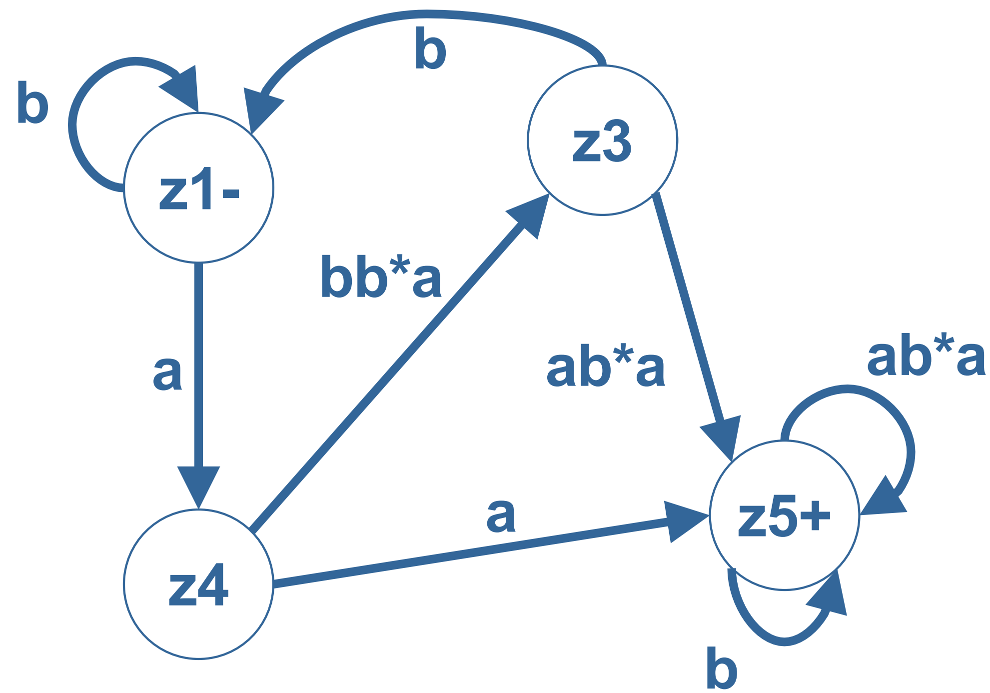
Complement(FA1’ + FA2’) = (FA1’ + FA2’)’

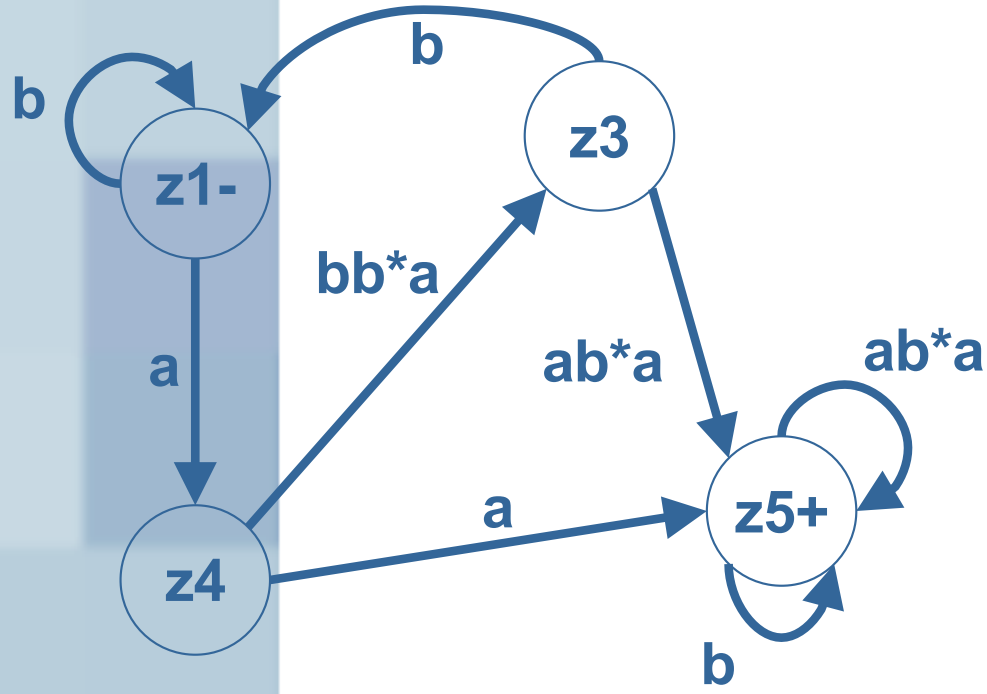


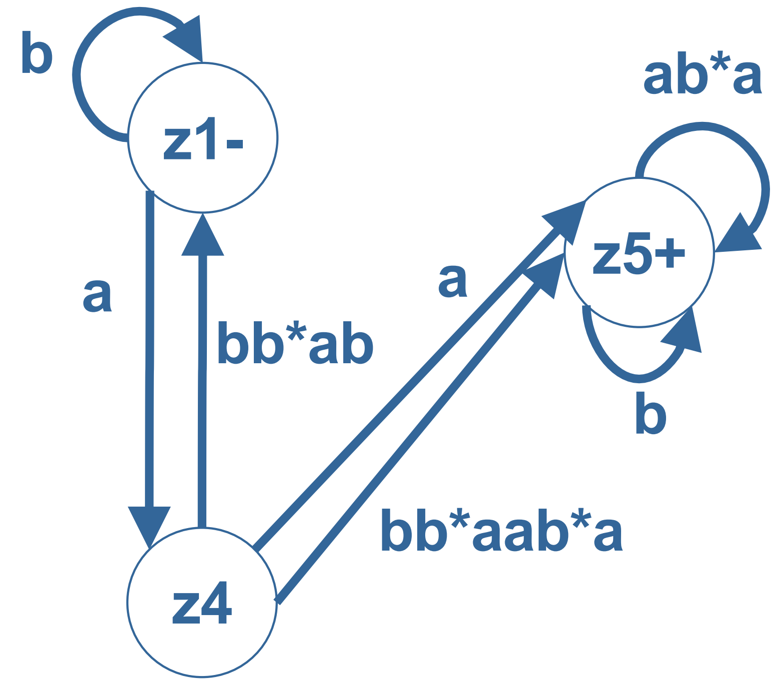

z1-
z2
z3
z4
z5+
z6
a
b

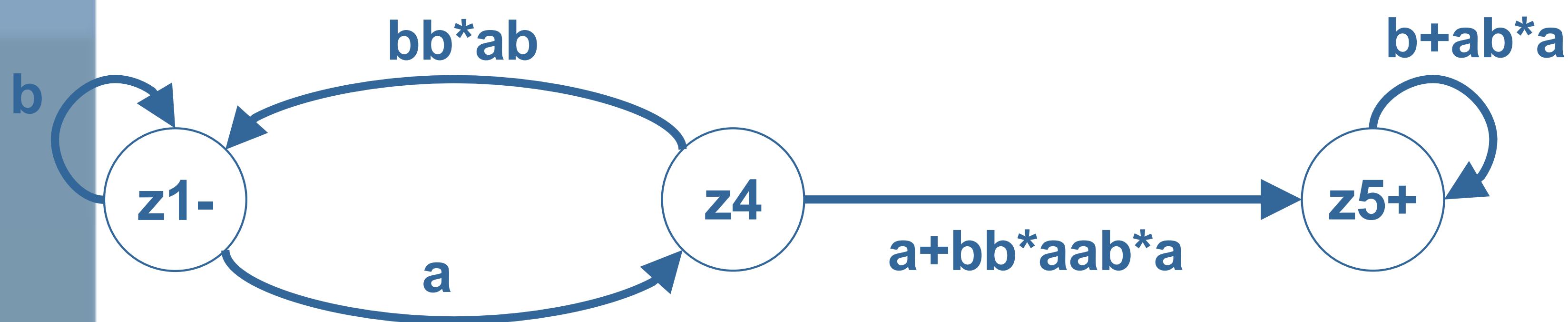

z1-
z3
z4
z5+
z6
bb*a
a
b

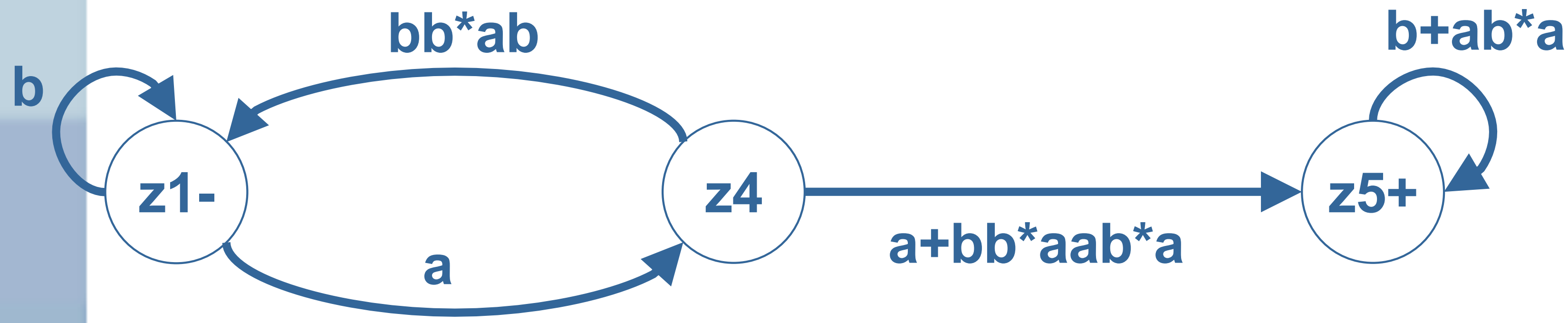

b
b
z3
z1-
a
bb*a
z6
a
b
a
b
a
z4
z5+
a

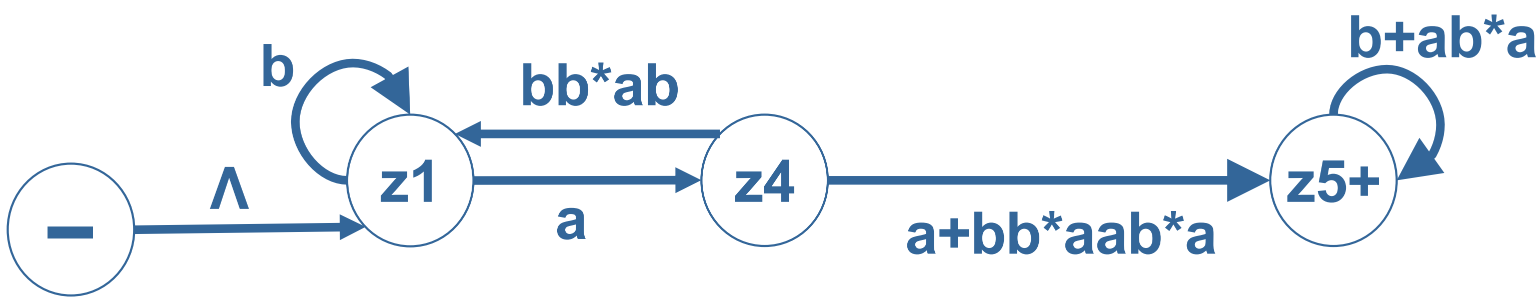

b
b
z3
z1-
bb*a
a
ab*a
ab*a
a
z4
z5+
b

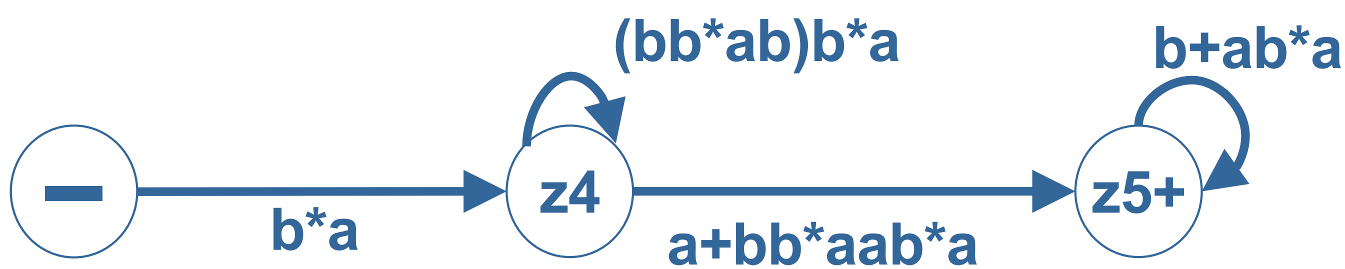

b
b
z1-
z3
bb*a
a
ab*a
ab*a
z4
a
z5+
b

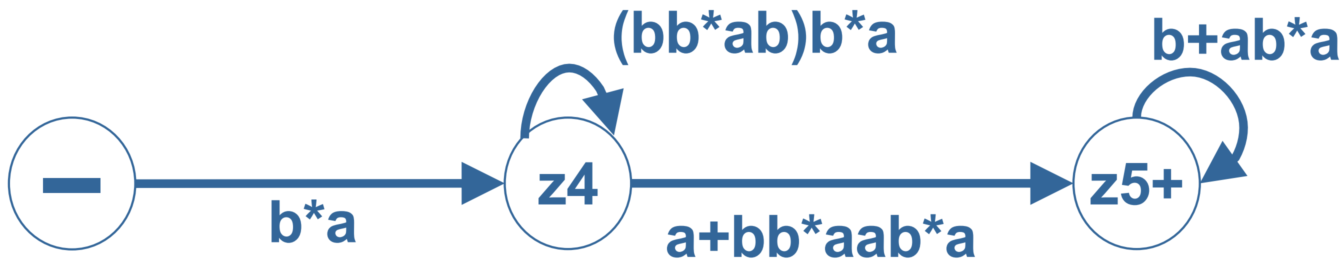

b
z1-
ab*a
z5+
a
a
bb*ab
b
bb*aab*a
z4

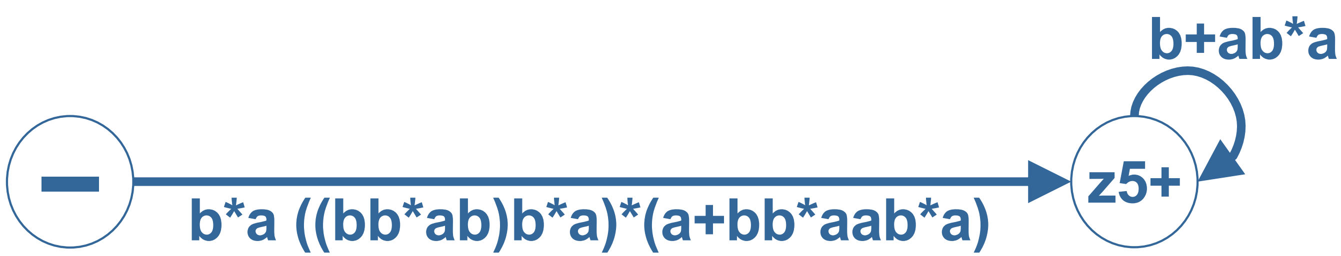

bb*ab
b+ab*a
b
z1-
z4
z5+
a
a+bb*aab*a


b
bb*ab
b+ab*a
z1-
z4
z5+
a
a+bb*aab*a


b
bb*ab
b+ab*a
Λ
z1
z4
z5+
a
a+bb*aab*a


(bb*ab)b*a
b+ab*a
z4
z5+
b*a
a+bb*aab*a





L1 ∩ L2 reduce to the regular expression

RE: b*a ((bb*ab)b*a)*(a+bb*aab*a) (b+ab*a)*